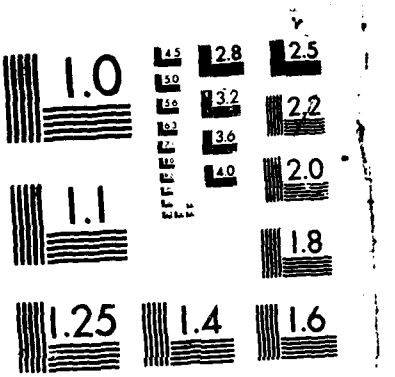
AD-A186 300 PARALLEL LOGIC PROGRAMMING AND ZMOB AND PARALLEL SYSTEMS SOFTWARE AND HAR (U) MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE J MINKER ET AL DEC 86 1/1

UNCLASSIFIED AFOSR-TR-87-1271 $AFOSR-82-0303 F/G 12/5 NL



UNCLASSIFIED

DTIC FILE COPY (2)

REPORT DOCUMENTATION PAGE

AD-A186 300

	1b. RESTRICTIVE MARKINGS
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT for public release, unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 87-1271
6a. NAME OF PERFORMING ORGANIZATION: Dept. of Computer Science	6b. OFFICE SYMBOL (If applicable):
6c. ADDRESS (City, State, and ZIP Code): University of Maryland	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
	7b. ADDRESS (City, State, and ZIP Code): AFOSR/NM [illegible] AFB DC 20332-6448
8a. NAME OF FUNDING / SPONSORING ORGANIZATION: AFOSR	8b. OFFICE SYMBOL (If applicable): NM
8c. ADDRESS (City, State, and ZIP Code): AFOSR/NM [illegible] AFB DC 20332-6448	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-82-0303
	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102F	2304	A7	

11. TITLE (Include Security Classification)

Parallel Logic Programming & ZMOB & Parallel Sys. Software & Hardware

12. PERSONAL AUTHOR(S)

Professor Jack Minker & Assoc. Professor Mark Weiser

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 6/30/82 TO 11/13/87	December 1986	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Under the current grant parallel hardware and systems software implemented on ZMOB in the previous year underwent extensive testing. A parallel problem solving system, PRISM (Parallel Inference System) implemented on the VAX/11-780 in the previous year was implemente on the PYRAMID and SUN machines.

DTIC ELECTE OCT 13 1987 S D E

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS		
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Rankin	22b. TELEPHONE (Include Area Code): (202) 767-5028	22c. OFFICE SYMBOL: NM

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

AFOSR-TR- 87-1271

Year-End Report

for

Parallel Logic Programming and ZMOB

and

Parallel Systems Software and Hardware

Sponsored by the

Air Force Office of Scientific Research

contract # AFOSR 82-0303

December, 1986

by

Professor Jack Minker

and

Assoc. Professor Mark Weiser

Department of Computer Science

University of Maryland

Abstract

Under the current grant parallel hardware and systems software implemented on ZMOB in the previous year underwent extensive testing. A parallel problem solving system, PRISM (Parallel Inference System) implemented on the VAX/11-780 in the previous year was implemented on the PYRAMID and SUN machines.

The initial version of PRISM uses a simulation of the ZMOB hardware, and has been fully tested and debugged. Experimental testing of PRISM on the simulated system was undertaken in the current year. In addition, several enhancements were made to PRISM to permit experimental analyses to be made, and to incorporate additional features to take full advantage of parallelism in a problem solving environment. The tracing and statistical gathering packages were extended. An AND-parallelism capability was added to achieve a second version of the PRISM system, and other features were added to the system to more fully exploit parallelism. A constraint solving machine was integrated with PRISM.

In addition to the above, a general method to permit informative answers to be presented to a user has been developed. Theoretical results were obtained for circumscription and a method for computing in "protected" circumscription, using Horn clauses was developed.

In the area of systems hardware and software, the ZMOB processor is now fully functional and in everyday use with 128 processors. Work is continuing on an experimental upgrade of some of ZMOB's processors to 68000s. Basic system software for multiprocessing on ZMOB is becoming more robust and performance studies now pinpoint areas for improvement. Studies of parallel software debugging continue to prove the value of multiple program views, and in particular the dicing approach was verified in a controlled experiment. We have also constructed an interactive visual slicer. Studies of the automatic parallelization of programs continues. We can now slice/splice arbitrarily structured programs and have techniques that significantly reduce information overhead between the slices and splicer.

1. Introduction

Under the current grant, a detailed design and implementation of a parallel problem solving system, PRISM (*p*arallel *i*nference *s*ystem*m*), based on logic previously achieved on the VAX/11-780 was transferred to the PYRAMID and SUN machines. PRISM was implemented using a simulated ZMOB belt. PRISM underwent experimental testing and was enhanced in a number of ways. The 128 processor ZMOB was made operational, as well as several forms of program slicers and splicers for automatic parallelization. Slicing received further experimental tests as a debugging tool, and new form of debugging based on "dicing" was invented. During the summer of 1986, PRISM was transferred to McMOB, the successor parallel machine to ZMOB.

In this report we provide a description of the accomplishments under the current grant. In the area of parallel problem solving, the initial PRISM has been fully implemented; individual programs have been implemented and tested in a parallel environment; and investigations have been made into extensions to the initial design. Experimentation has begun on evaluating PRISM in the simulated environment. We have developed extensions to circumscription and to computing in the case of protected circumscription. An approach has been developed to providing informative answers to users of deductive databases and problem solving systems. In the area of parallel systems hardware and software, the ZMOB is fully functional with 128 processors and in daily use; the McMOB 16 processor successor to ZMOB is opeartional and in daily use; systems software is reliable; and several advances have been made in the areas of automatically parallelizing programs and understanding their debugging. As a consequence of the work, two book chapters on Zmob have been solicited, three journal articles have appeared, eleven papers were accepted for publication in refereed conference proceedings, three PhD theses were written, three MS scholarly papers, and six technical reports. One article has been submitted to a journal, a second journal article is in progress, several papers will be appearing in refereed conferences, and other papers are in progress.

We believe that there is currently no comparable effort that exploits parallelism in logic programs. Although the current research is oriented towards moderate size problems, the attachment of disks and backend relational database machines will permit large problem solving and relational databases to be

executed in the ZMOB environment.

The research in parallel problem solving is under the direction of Professor Jack Minker who is directing the current parallel problem solving efforts on ZMOB. Dr. Don Perlis whose research has been in non-monotonic logics and artificial intelligence has been working with Dr. Minker on some of the theoretical issues associated with non-monotonic logic and parallel logic programs. The parallel hardware and software effort is under the direction of Associate Professor Mark Weiser.

2. Accomplishments on Effort During Period December, 1985-December, 1986

This section is subdivided into two major parts. The first section, 2.1, describes the accomplished research with respect to PRISM - the parallel problem solving system. The second section, 2.2, describes the efforts for the development of systems software and hardware to make the ZMOB system available as a resource for experimentation with parallel algorithms.

2.1. PRISM and Parallel Problem Solving on ZMOB

There were eleven major tasks in parallel problem solving undertaken under the current grant. These are:

(a) Enhancements to PRISM:

(1) Integrate the Constraint Solving Machine with PRISM

(2) Implement Independent and Dependent AND-Parallelism

(3) Implement new tracing and debugging facilities for PRISM

(4) Interface PRISM with a relational database system

(5) Implement the Intelligent Channel

(b) Evaluation Studies

(1) Empirical Studies on PRISM

(c) Theoretical Studies:

(1) Control Structure Investigations

(2) Investigation of Typed Logics

(3) Non-Monotonic Logic Investigations

(4) Intelligent and Cooperative Answers

(5) Investigation of Parallel Computation

During the current grant substantial work was accomplished for all but two of the listed tasks. The PRISM system was enhanced by the addition of AND-parallelism, the integration of the Constraint Solving Machine, and the addition of new tracing facilities. Work was accomplished in the area of theoretical studies. Papers were published in the areas of Non-Monotonic Logic, Intelligent and Cooperative Answers and Parallel Computation.

2.1.1. Enhancements to PRISM

The initial PRISM system was enhanced by the addition of many features. The features and their contributions to the system will now be outlined.

2.1.1.1. Integration of Constraint Solving Machine (CSM)

During the current grant year the CSM was integrated with the PRISM system. This effort involved modifying the Problem Solving Machine (PSM) so that it sends goals to be tested for consistency to the CSM. The system was tested on numerous databases including a genealogy database and a database that represents a typed logic using a representation that has a semantic network flavor. Statistics on the performance of the system were gathered. Results show that the CSM can successfully be used to reduce the search space in problem solving in much the same way that the use of a typed logic can. Future work in this area involves the implementation of a full theory of constraints as in [Kholi Portugal].

2.1.1.2. Implementation of AND-Parallelism

We have implemented and tested a version of PRISM that can exploit both full AND-parallelism and independent AND-parallelism. The PRISM system is now such that it can run with either OR-parallelism alone, OR-parallelism with independent AND-parallelism only, and OR-parallelism with both dependent and independent AND-parallelism. In addition to these modes the user can control the use of these types of parallelism through the use of annotations on the given logic program. Through the use of these features the user has flexible control over how the system executes goals in parallel.

We note that the features outlined above help make PRISM one of the most advanced existing parallel logic programming systems. No other system currently has the ability to exploit these forms of parallelism in such a flexible manner.

2.1.1.3. Implementation of Tracing and Debugging Facilities

Two types of tracing facilities were added to PRISM under the current grant. The first allows the user to see how PRISM constructs derivations. This system defines a window for each Problem Solving Machine (PSM). The window contains the current proof tree that is active in the PSM and displays information about the messages that it receives from other machines. Currently in order to use this system the trace information is collected online and the trace is run afterwards. However, now that equipment for graphics is more readily available it will be relatively easy to make this trace run online.

The second type of trace allows the user to examine offline the structure of a proof tree collected by PRISM. The user is allowed through the use of keyboard commands to walk up and down the proof tree. At each node the user can ask for information about the node's status. The information includes such things as the literal selected in the node, the unifier and resolvent of the node and whether or not the node was sent to another Problem Solving Machine.

Each of the above facilities greatly enhance the user's ability to understand and debug PRISM programs. The first system allows the user to understand how PRISM passes messages and exploits parallelism. The second allows the user to find and isolate bugs in the logic of the program.

2.1.1.4. Other Enhancements

Due to lack of time we did not work on interfacing PRISM with a relational database system or implementing the Intelligent Channel concept. We still believe that it would be very useful to integrate both of these features into PRISM.

2.1.2. Evaluation Studies

During the current grant year the individual PRISM modules were instrumented so that they could collect runtime statistics. A series of experiments were run trying to determine information about how the allocation of PRISM machines effect the running of the system. There were two basic experiments:

(1) The first involved determining whether a speed up in reasoning occurs when additional machines are used in problem solving.

(2) The second experiment involved trying to characterize how PRISM machines should be allocated between Problem Solving Machines and Database Machines when only a fixed number of machines are available.

These experiments were run on the following databases in both OR-parallel mode and, Independent AND-parallel and OR-parallel mode.

(1) A genealogy database

(2) The four queens problem

(3) A quad tree program

(4) A natural language parser

(5) A program for determining that the fringes of a tree are equivalent.

The results obtained were as follows:

(1) When the encoding of the problem was designed for running in an OR-parallel or AND-parallel mode then a linear speed up in runtime occurs when additional machines are used in problem solving until some asymptote is reached.

(2) For all databases when given a fixed number of machines they should be divided approximately equally between Problem Solving Machines and Database Machines.

The first result is valuable in that it suggests that we can exploit parallelism only if we can learn how to encode problems in a manner that exploits parallelism. The fact that the system works best when there is one database machine for every problem solving machine seems to suggest that there is no need to split the machines up. However, a deeper analysis of the results show that performance would decay if the PSMs did not have the ability to send queries to different database machines. These preliminary results must be verified with experiments on McMOB.

2.1.3. Theoretical Studies

2.1.3.1. Control Structure Investigations

An initial version of a control structure compiler has been written and tested. The compiler takes a specification of the control component of a logic programming language and creates an efficient interpreter for the language. The compilation technique is important in that it allows the user to explore issues in the control of logic programs without having to resort to the use of a meta-interpreter. There are two primary advantages to this approach. First, one does not pay the cost in terms of speed that one must pay when using the meta-interpreter approach. Second, when writing a meta-interpreter one has to fight with PROLOG's depth first with backtracking strategy. Since the compiler system is not constrained in such a manner, it is easier for the user to conceive and express a control strategy.

Currently interpreters for the following control strategies have been generated:

(1) Depth First, Left to Right Literal Selection

(2) Breadth First, Left to Right Literal Selection

(3) Depth First, Selects Most Instantiated Literal

(4) Breadth First, Selects Most Instantiated Literal

Initial timing results show that the interpreter that mimics PROLOG (1) does so with a factor of two loss of speed.

2.1.3.2. Investigation of Typed Logics

Implementation strategies for typed logics were discussed during the year. As stated in the section on the Constraint Solving Machine it was determined that the CSM can be used to express most of the desired type information. Other strategies considered involved making modifications to the unification algorithm.

2.1.3.3. Non-Monotonic Logic Investigations

In the previous year we investigated McCarthy's notion of formula circumscription (McCarthy [1984]) and found a partial completeness result for formula circumscription. We also investigated computational issues associated with protected circumscription and developed an algorithm that extends the con-

cept of relational databases and permits complete and sound answers to be found in the case where the theory consists of ground atomic formulae extended to contain protected data. We then investigated the general case of Horn deductive databases augmented with protected data and found that the obvious extension of our algorithm to this case was not adequate. Two journal articles were written in this year based on these results.

2.1.3.4. Cooperative Answers for Database Queries

In this research we show how to utilize the semantics already present in a database in the form of integrity constraints to give cooperative answers to database queries. A natural language interface to a database has been designed. The interface is composed of the following modules:

(a) A natural language parser which transforms natural language queries into a logic formula.

(b) An optimizer which receives a logic formula applies integrity constraints to optimize the query and collects all possibly useful integrity constraints concerning the query.

(c) A query evaluation procedure which outputs a useful answer in natural language.

Most of the current work has focused on the third module. The work achieved by this module can be divided into two components, "What to Say" and "How to Say it". The "What to Say" problem is guided by the heuristic use of the integrity constraints found in the second module. The "How to Say it" problem is addressed by using the inverse of the first module. As a result of this research two papers were presented at conferences.

2.1.3.5. Investigations in Parallelism in Graph Theory

A powerful framework for developing efficient solutions to graph theoretic problems has been devised and used to solve many problems. The framework is based upon the idea that graph separator theorems can be used in formulating efficient divide- and-conquer solution strategies. A separator theorem provides an efficient method of partitioning graphs into two components by removing only a few edges. Many classes of graphs including planar graphs, series-parallel graphs and outerplanar graphs are known to posses good separator theorems. Problems solved using this method include, efficient embeddings of graphs into binary trees and area efficient VLSI layouts for graphs of arbitrary degree. A paper describing

these results has been submitted to the Journal of the ACM.

Incremental parallel algorithms for a class of graph problems based on spanning trees have been developed. The machine model for these algorithms is the parallel random access machine (PRAM). These algorithms exhibit a speed up of log(n) over the corresponding algorithms that solve these problems from scratch. These algorithms are versatile in that they can be run efficiently on all three models of PRAMs (viz. Concurrent Read and Write, Concurrent Read and Exclusive Write, Exclusive Read and Write). The problems solved include connected components, bridges and bridge connected components, cycle basis and minimum spanning trees. Early results of this work appear in the proceedings of the Fifth Conference on Foundations of Software Technology and Theoretical Computer Science. A comprehensive account of these results can be found in Maryland technical report TR-1590.

2.2. Parallel Hardware and Software

The parallel hardware and software research under this grant has the following goals:

(a) Construction and evaluation of the Zmob architecture.
(b) Construction and evaluation of a variety of systems software for Zmob.
(c) Investigation of languages and operating systems for parallel computation.
(d) Investigation of general strategies for parallel computation.

Work has been proceeding in all four areas, although (a) and (b) have necessarily dominated the early phases of research. There has been good progress, and research has now turned more to (c) and (d) and investigating more general concerns of parallelism as guided by the Zmob experience.

2.2.1. Construction and evaluation of Zmob

At the time of our last report the ZMOB was running with only 64 processors. It is now running with a full 128 processors, is in daily use for a variety of research projects (primarily systems and numerical analysis).

The 128 processor ZMOB continues to operate with a clock rate of 6.5Mhz, reduced from the design speed of 10Mhz. A 16 processor ZMOB has been made to run at the 10Mhz rate, and the lessons learned from this will be incorporated into the larger processor this year. The slower clock forces software to use the slower programmed I/O on the Z-80A rather than the much faster DMA on the ZMOB processor

boards. Thus the faster clock will speed up parallel communication by even better than the apparent 1.5 factor.

We constructed a 16 processor 68010 Zmob, called McMOB, in wire-wrap form, and this became available for dedicated research work during the late Summer of 1986. McMOB's uses the MOB communications boards and message-passing strategy, but, instead of Z-80 processors with 64kbytes of memory, the McMOB processors are 68010's with 1Mbyte of memory.

Among the special hardware purchased to support ZMOB research this year were several disk drives. The disk drives are being integrated into the MacMOB 68010 processors, which will also have direct ethernet connections soon.

2.2.2. Construction and evaluation of systems software

Any new architecture needs considerable attention to its underlying support software before it can be used. This support software includes debuggers, cross-compilers, assemblers, 'tweakers', linkers, loaders, simulators, etc. This software exists for ZMOB in abundance, and is in a fairly stable state. In fact, the majority of it achieved a milestone this year by moving from research status and hence the responsibility of this grant to 'supported' status and hence a standard part of the Maryland CS Department Laboratory software, supported by the systems staff. Previous proposals and reports have dwelt on this systems software, but will do so no longer. No more research: it is there, use it.

The evaluation of this systems software is now well underway. There are several projects to measure the overhead of communicating and programming on ZMOB, the net result of which at the moment is that one is better off in assembler language with the current ZMOB. The main reason is the mismatch of the C language with the Z-80 and ZMOB architecture, which is due in part to the lack of DMA at the current clock rate. We have also measured, using the current software, essentially linear speed-up in certain numerical analysis problems as the number of processors goes from 1 to 63. This is an encouraging, if problem dependent, result. We certainly plan more work along these lines.

Another important systems event was the development by us of support in the Unix 4.2bsd kernel for the Xerox XNS protocol suite. This is important for investigating parallelism over networks other

than ZMOB and for comparison with ZMOB. This implementation was reported (O'Toole et al [1985]) and since then has been distributed to several other universities.

2.2.3. Investigation of general strategies for parallelism

We have several theoretical results in the area of general strategies for parallelism. In previous years we developed a theory of program slicing which describes how to produce program subsets which model projections of a program's behavior. These subsets could be run in parallel, and in principle contained enough information to reconstruct their original program's behavior. The proof of this principle required the theory of 'splicing', in which it was shown that for structured programs this behavior reconstruction could be done in real-time.

We have been extending the power of theoretical analyses in several ways. First, we have represented splicing in graph theoretic terms. Second, we have extended the splicing theory to general programs and their graphs , not just structured programs. Third, if one is interested in a projection of the original program's behavior, say just the relative ordering of the output statements, this work presents a technique that significantly reduces the amount of information that has to be spliced. Finally, we have new definitions and proofs which are much more elegant than the original language-theoretic approach. A paper on this work has been submitted to Acta Informata. We expect it will be of interest in the field of parallel programs as well as in the fields of graph theory and abstract automata, to which the results generalize.

A long standing problem in slices is handling program input. Earlier work requires that all input data be sent to all slices, with the possibility that the data is simply ignored by most of them. This carries the communication cost of moving that data around, and the slice run-time cost of reading it in, possibly in a loop with its associated overhead. We refer to this problem as the "splitting" problem. We now have a technique that enables a slice to "bypass" the input of an input statement, if it cannot possibly affect (as detectable by data flow analysis) its computation.

2.2.4. Practical Slicing and Splicing

In order to give the automatic slicing/splicing theory of parallelism a good empirical test we are constructing a "production quality" slicing/splicing compiler. The goals of this compiler are to automatically parallelize large Fortran programs, to be compatible with existing low-level parallelization efforts (such as vectorization), and to be invisible to the programmer. We achieve goal one by accepting the full f77 language as embodied in the Unix f77 compiler. We achieve the second goal by slicing and splicing completely at the source code level, thereby permitting a subsequent optimization phase to work on our sliced code. The third goal is the research challenge, but is well on its way to being achieved for procedureless programs. We are incorporating procedures now, and expect to have a first complete prototype of the entire system done by fall.

3. Bibliography and References

(1) Cao, D., "Design of the Intensional Database System of the ZMOB Parallel Problem Solver", Technical Report, Computer Science Department, University of Maryland, 1982.

(2) Chakravarthy, U.S., "Semantic Query Optimization", Ph. D. Thesis, Department of Computer Science, University of Maryland, 1985.

(3) Chakravarthy, U.S., Fishman, D., and Minker, J., "Semantic Query Optimization in Expert Systems and Database Systems", University of Maryland, July 1984.

(4) Chakravarthy, U.S., Minker, J. and Tran, D., "Interfacing Predicate Logic Languages and Relational Databases". *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, Faculte des Sciences de Luminy Marseille, France, 91-98.

(5) Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach", *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, September 14-17,1982, Faculte des Sciences de Luminy Marseille, France, 71-77.

(6) Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach". Technical Report TR-1124, Computer Science Department, University of Maryland, December 1981.

(7) Futo, I., "A Constraint Machine to Control Parallel Search on Prism", Department of Computer Science, University of Maryland, 1984.

(8) Gal, A. and Minker, J. "A Natural Language Database Interface Giving Cooperative Answers", Department of Computer Science, University of Maryland, 1985.

(9) Gallaire, H., Minker J. and Nicolas, J-M., "Logic and Databases: A Deductive Approach", Computing Surveys, Vol. 16, No. 2, pp. 153-185, June 1984.

(10) Gallaire, H., Minker, J. and Nicolas, J.-M., "Advances in Data Base Theory, Volume 2", Plenum Publishing Company, February, 1984.

(11) Kasif, S. and Minker, J. "The Intelligent Channel: A Scheme for Result Sharing in Parallel Logic Programs", Proceedings International Joint Conference on Artificial Intelligence, August 1985.

(12) Kasif, S., "Analysis of Parallelism in Logic Programs", Ph. D. Thesis, Department of Computer Science, University of Maryland, December 1985.

(13) Kasif, S., Kohli, M., and Minker, J., PRISM: A Parallel Inference System for Problem Solving, Proceedings of the 8th International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany, 544-546.

(14) Kasif, S., Kohli, M., and Minker, J., "PRISM - A Parallel Inference System Based on Logic", Technical Report TR-1243, Computer Science Department, University of Maryland, February 1983.

(15) Kasif, S., Kohli, M., and Minker, J., "PRISM: A Parallel Inference System for Problem Solving", Proceedings Logic Programming Workshops, Praiada Falesia, Algarve, Portugal, Universidad Nova De Lisboa, 26 June - 1 July 1983, 123-152.

(16) Kohli, M. "Controlling the Execution of Logic Programs", Proposed Ph. D. Thesis, Department of Computer Science, University of Maryland, May 1986.

(17) Kohli, M., and Minker J., "Control of Logic Programs Using Integrity Constraints" Proceedings Logic Programming Workshop 1983, 26 June - 1 July 1983, 123-152.

(18) Kohli, M., and Minker, J., "A Theory of Intelligent Forward and Backward Tracking", Technical Report (in preparation), Computer Science Department, University of Maryland, March 1983.

(19) Kohli, M., and Minker, J., "Intelligent Control Using Integrity Constraints", Proceedings of the National Conference on Artificial Intelligence, August 22-26, 1983, 202-205.

(20) Lyle, J. Evaluating Variations on Program Slicing for Debugging. Ph.D Dissertation, Computer Science Dept., University of Maryland. December 1984.

(21) Lyle, J. and Weiser, M. Evaluating variations on program slicing for debugging. in preparation, 1985.

(22) Minker, J. Perlis, D. and "Completeness Results for Circumscription", Department of Computer Science, University of Maryland, January 1985.

(23) Minker, J. and Perlis D., "Applications of Protection of Circumscription", Proceedings of the Conference on Automated Deduction- , Napa, California, March 1984.

(24) Minker, J., and Perlis, D., "On The Semantics of Circumscription", Technical Report 1341, Computer Science Department, University of Maryland, August 1983.

(25) Minker, J., et al., "Functional Description of the ZMOB Parallel Problem Solving System", Technical Note 1, Department of Computer Science, University of Maryland, December 1982.

(26) Minker, J., et al.,"Parallel Problem Solving on ZMOB", Proceedings of Trends and Applications 83, Washington, D.C.,1983.

(27) O'Toole, J., Torek, C., and Weiser, M. Implementing XNS protocols for 4.2bsd. Usenix Unix Users Conference, Dallas TX. January 1985.

(28) Weiser, M., Kogge, S., McElvany, M., Pierson, R.. Post, R., and Thareja, A. Status and Performance of the Zmob Parallel Processing System. IEEE CompCon conference, San Francisco, CA, February 1985.

END

DATE

FILMED

JAN

1988